Models of Software Systems
Qualifying Examination

July 2015

1	/ 40
2	/ 60
Total	/ 100

4. (40 pts) State Machine

The following is a description of a Simple Infusion Pump.

> Set of actions that can be selected interactively to this machine are:
> "plug_in", "set_value", which is written':=', "reset", "fill_fluids"
> Pump starts in power off state.
> User must plug pump in before anything else can happen.
> Before pump operation starts, user must enter amount of medicine to deliver to patient.
> Main operation of this machine is "pump".
> User may reset pump at any time
> When the pump has delivered the amount of medicine requested it goes to the DONE state.
> When fluid runs out, the pump goes into an alarm state.
> Otherwise, the pump delivers one unit of medicine
> Error state associated with empty pump:
> Repeatedly rings bell until user refills the pump

Model the Simple Infusion Pump with a state machine. To model accurately, you may declare variables of various types and use Boolean conditions as "guard" conditions such that only when the guard is true, the action associated with it can be performed. Also you may use input and output actions. In general, you may use any and any combinations of the following four forms of action descriptions to make the state machine satisfy the above requirements:

i) *action*
ii) *input action* / *output action*
iv) [*guard*] *action*
iii) [*guard*] *input action* / *output action*

3. (60 points) State Machines and Temporal Logic

Consider a simple state machine model of an elevator system with three floors. Assume that the elevator has two state variables: *floor*, which represents which floor the elevator is on; and *requested*, which represents which floors have been requested for stops by users. When the elevator attempts to service a floor request, it visits all (and only those) intermediate floors that have outstanding requests.

The values for *floor* are one of the following:

- *On*1: The elevator is on floor 1;
- *On*2*Up*: The elevator is on floor 2 and will give priority to requests to go to the third floor before requests to go to the first floor;
- *On*2*Down*: The elevator is on floor 2 and give priority to requests to go to the first floor before requests to go to the third floor; and
- *On*3: The elevator is on the third floor.

The values for *requested* are *sets* of zero or more of the values f_1, f_2, and f_3 (representing requests for floor one, two and three, respectively). Values are added to the set when a user presses one of the elevator buttons (either inside the elevator, or on a floor), and are removed when an elevator visits the floor requested.

1. (12 points) This model abstracts away many aspects associated with a real elevator. List three aspects of a real elevator that are missing from this model.

2. (48 points) (12 points for each of (a)~(d)) For each of the following, (i) express the claim in linear temporal logic, and (ii) say whether it is true of this model. (You may use floor values as predicates. For example, *On*1 can be used as a predicate to indicate that the elevator is on floor 1. Similarly, you may use f_i as a predicate to indicate that floor *i* has been requested.)

   (a) The elevator visits floor 1 only finitely many times.

   (b) In all traces if f_2 is pressed, eventually f_2 will be visited.

   (c) The elevator will never go directly from the *On*1 state to the *On*2*Down* state.

   (d) The elevator will remain on floor 1 until it moves from floor 1 to floor 2.